**Introduction to Design**

Design is the process in which user requirements are transformed into some useful forms which further helps a programmer in Coding or Implementation phase. Design is done in software development because it helps in preventing redundancy increases reusability and it is the best approach to mitigate the risk that we have no idea about.

**Structural Diagram**

**Final Class Diagram:** Class diagram in UML (Unified Modelling Language) is a static diagram that describes the system structure by demonstrating the system classes, attributes, operations and relationship that exists among the objects.

**Justification for the approach taken**

For following reasons, I have made use of class diagram:

1. It shows static structure and classifiers in a system.
2. It is useful for developer as well as team member too because this diagram shows different class and object and relationship that exists between them.
3. It also describes the functionalities done by the system.

**Notations Used**

**Class**



**Attribute**



**Generalization**



**Association**



**Multiplicity**



**Aggregation**



**Final Class Diagram**

**Description of Diagram**

**Data Flow Diagram:** It is a diagram that models the flow of information for any information system or process. This modelling techniques helps to show the process that are involved in a system to make transfer of data from the input to the database (File Storage) and report production.

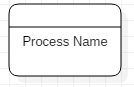
**Justification for the approach taken**

DFD helps in representing process or functions which manipulate, captures, stores and allocate data between the systems. For following reasons, I have used DFD diagram:

1. It helps in describing the logical flow of system.
2. This diagram uses simple notation so it easily understandable.
3. It also helps in creation of automated and manual system requirements.

**Notations Used**

**Process**



**Database**



**Data Flow**

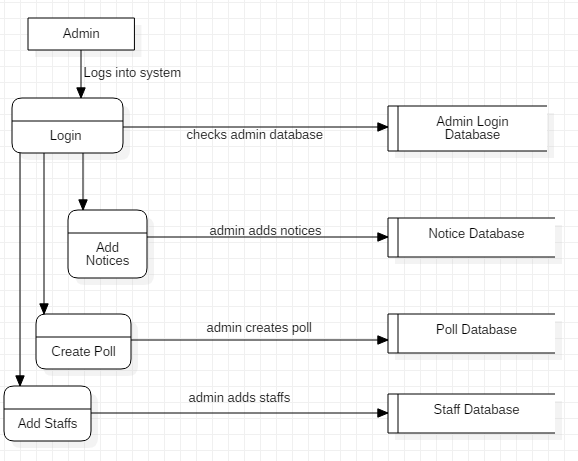


**External Entity**

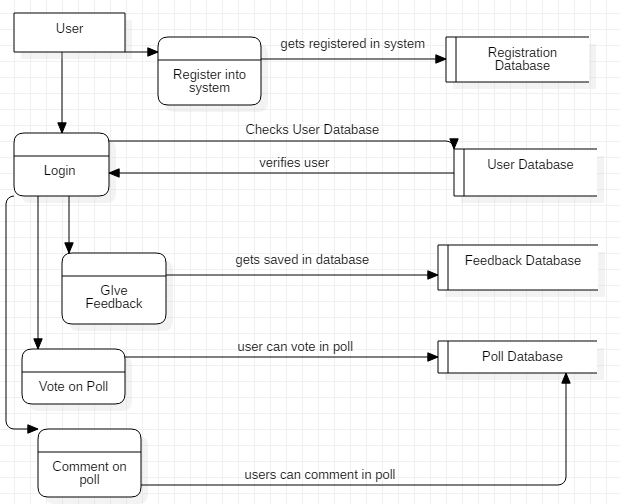


**Data Flow Diagram**

**For Admin**



**For User**



**Description of Diagram**

In optional diagram I have made use of data flow diagram. I have designed two DFD for admin as well as user. In DFD process done and datastore where the data are stored is modelled and these are shown with the help of Control flow.

**Behavioral Diagram**

**Activity Diagram:** It is an important diagram that models the dynamic aspects of the system. It is a kind of flowchart that helps to represent the flow from one activity to another activity. The flow can be sequential, concurrent or branched.

**Justification for the approach taken**

For Following reasons, I have made use of activity diagram:

1. It helps in describing the activity flow of system.
2. It describes the sequence among different activity.
3. It defines branched, concurrent, sequential flow of system.

**Notations Used**

**Start Point/ Initial**



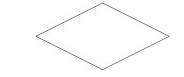
**Activity**



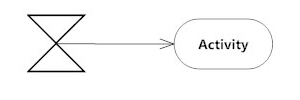
**Action Flow**



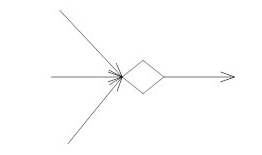
**Decision**



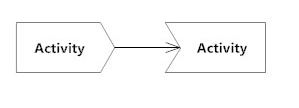
**Time Event**



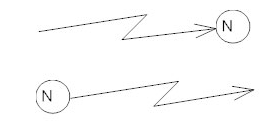
**Merge**



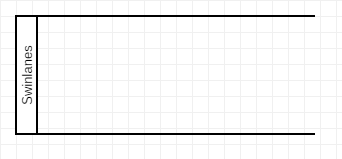
**Message sent and received**



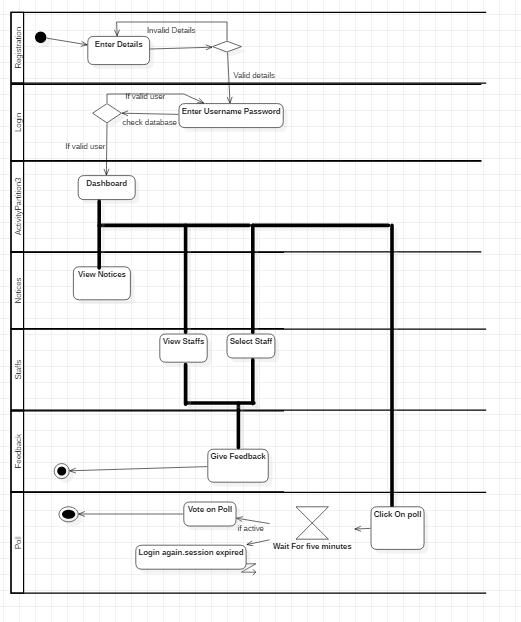
**Interrupting Edge**



**Swim lanes**



**Activity Diagram**



**Description of the Diagram**

Activity diagram is used as behavioral diagram and I have used this diagram as to describe the activity flow of the system. All activity that can be done by user are modelled in this diagram. Also, time sequence, initial, end phase and interrupts are shown in this diagram. Decisions node and what decisions are to be made upon different conditions are shown in this diagram.

**Sequence Diagram:** Sequence diagram are interaction diagrams that represents how different operations are carried out. They help to capture interaction among and between objects in situation of a collaboration.

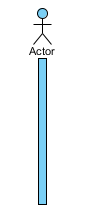
**Justification of approach taken**

For following reasons, I have made use of sequence diagram:

1. It models high-level interaction between objects that are active in a system.
2. It shows the interaction between objects within relationship that realizes a process.
3. It helps in understanding and the detailed functionality of an existing or upcoming scenario and also helps to plan them.

**Notations Used**

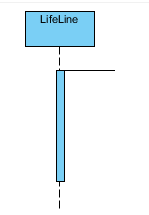
**Actor**



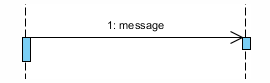
**Lifeline**



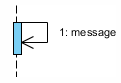
**Activation**



**Call Message**

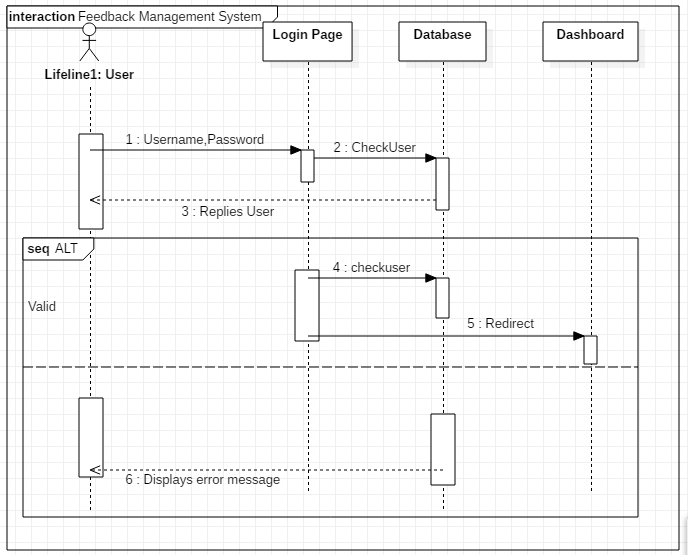


**Self-Message**

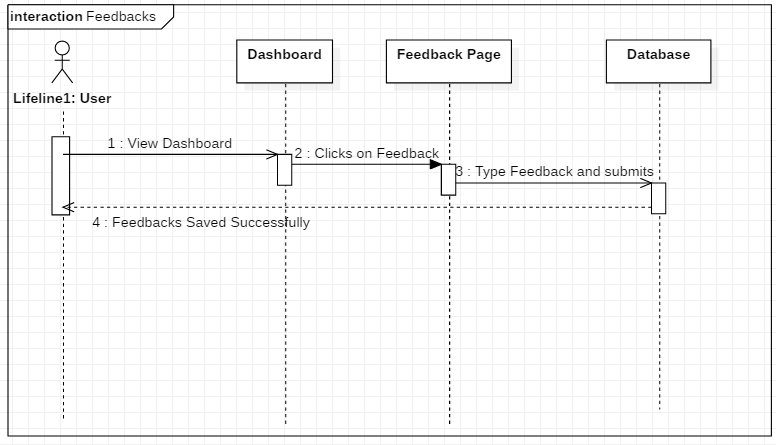


**Sequence Diagram**

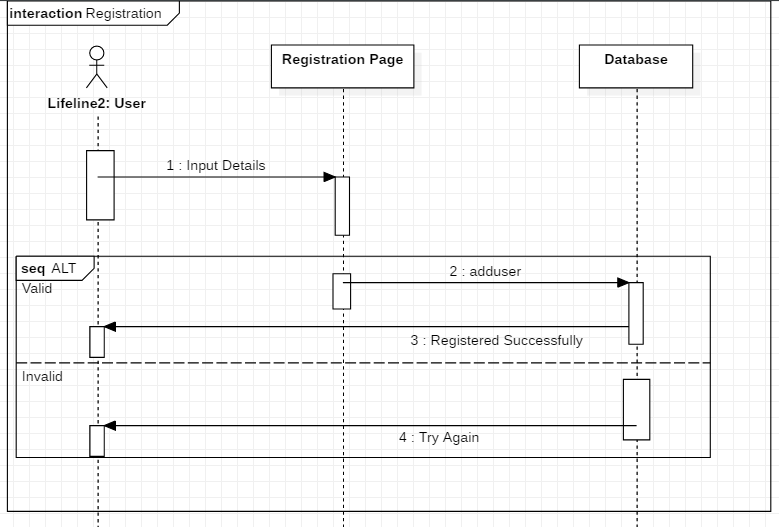
**For Login**



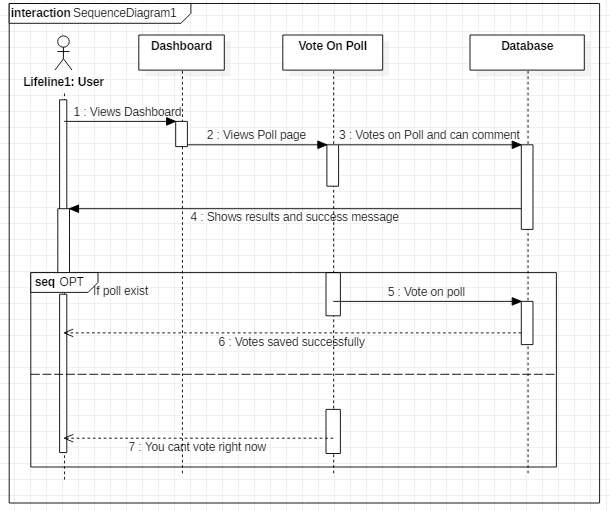
**For Feedback**



**For Registration**



**For Voting on Poll**



**Description of diagram**

I have made four different main Sequence diagrams for this project. I have made sequence diagram for Login, Registration, Feedback and Vote on poll which are vitals of my system. Above diagram shows on how operations are carried out. Also, it shows when an object is active and signals received/sent by it. Alternate and optional sequence diagrams are also shown in above diagram.

**Database Modelling**

**Data Dictionary:** Data dictionary is a type of table which describe the contents, structure, format and the relationship that exists among the elements. They can be used for manipulating the database and controlling access. The users of database don’t really have access to data dictionary it is all managed by Database Administrator.

Data dictionary for my database is given below:

**User**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | UserID | Int | 04 | Not null | Primary Key | Pk\_userid |
| 2 | First Name | Varchar | 20 | Not null | - |  |
| 3 | Last Name | Varchar | 20 | Not null | - |  |
| 4 | Email | Varchar | 50 | Not null | - |  |
| 5 | Department | Varchar | 30 | Nullable | - |  |
| 6 | Username | Varchar | 20 | Not null | - |  |
| 7 | Password | varchar | 10 | Not null | - |  |

**Department**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | DepartmentID | Int | 04 | Not null | Primary Key | Pk\_dept\_id |
| 2 | Department Name | Varchar | 55 | Not null | - | - |

**Staff**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | StaffID | Int | 04 | Not null | Primary Key | Pk\_Staff\_Id |
| 2 | First Name | Varchar | 20 | Not null | - |  |
| 3 | Last Name | varchar | 20 | Not Null | - |  |
| 4 | Email | varchar | 30 | Not null | - |  |
| 5 | Qualification | Text | - | Not null | - |  |
| 6. | Comment | Text | - | Nullable | - |  |
| 7 | Date of Join | Date | - | Not null | - |  |
| 8 | DepartmentID | int | 04 | Nullable | Foreign Key | Pk\_dept\_id |

**Rating**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | UserID | Int | 04 | Not null | Foreign Key | fk\_user\_id\_ |
| 2 | StaffID | Int | 04 | Not null | Foreign key | Fk\_staff\_id\_ |
| 3 | Rating | Int | 05 | Not Null | - | - |

**Poll Create**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | PollID | Int | 04 | Not null | Primary Key | Pk\_poll\_Id |
| 2 | Question | Varchar | 20 | Not null | - |  |
| 3 | Option1 | varchar | 100 | Null | - |  |
| 4 | Option2 | varchar | 100 | Not null | - |  |
| 5 | Posted\_date | Date | - | Not Null | - |  |

**Vote on Poll**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | PollID | Int | 04 | Not null | Foreign Key | fk\_poll\_Id |
| 2 | UserID | Varchar | 20 | Not null | Foreign Key |  |
| 3 | Option | Data Type |  | Not Null | - |  |
| 4 | Comment | Text | - | Nullable | - |  |

**Feedback**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | FeedbackID | Int | 04 | Not null | Primary Key | Pk\_feedback \_Id |
| 2 | UserID | int | 04 | Not null | Foreign Key | Fk\_usr\_id |
| 3 | Feedback | text | - | Not Null |  |  |
| 4 | StaffID | Int | 04 | Not null | Foreign Key | Fk\_stffs\_ |
| 5 | Posted\_date | Date | - | Not Null | - |  |

**Question**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | QuestionID | Int | 04 | Not Null | Primary Key | Pk\_question\_id |
| 2 | UserID | Int | 04 | Not Null | Foreign Key | Fk\_usd\_ |
| 3 | Question | text | 255 | Not Null | - | - |
| 4 | Date | Date | - | Not Null | - | - |

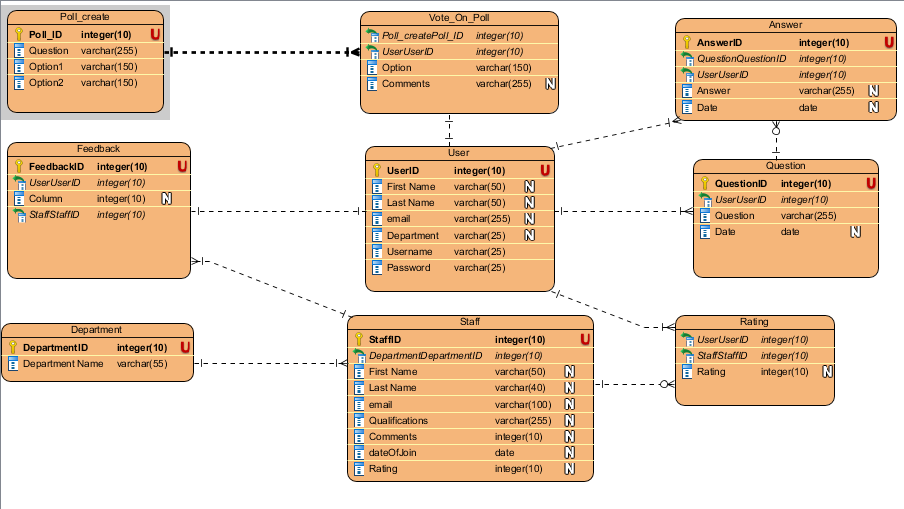
**Answer**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Column Name | Data Type | Length | Null | Key | Constraint |
| 1 | AnswerID | Int | 04 | Not Null | Primary Key | Pk\_question\_id |
| 2 | QuestionID | Int | 04 | Not Null | Foreign key | Fk\_question\_id |
| 3 | UserID | Int | 04 | Not Null | Foreign Key | Fk\_usrid\_ |
| 4 | Answer | text | 255 | Not Null | - | - |
| 5 | Date | Date | - | Not Null | - | - |

**ER Diagram**

It is aa structural diagram used in database design which make uses of various symbols and connectors that shows the major entities within the system scope and the inter-relationship between objects. ER diagram are straightforward and shows relationships between entities in clear way and ER diagram can also be easily converted to other data model.

ER diagram for my database is given below:

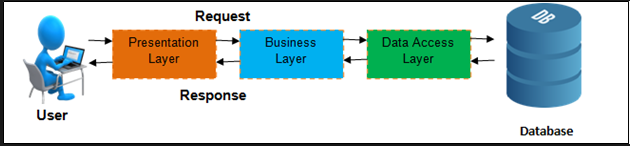


**Architectural Model**

I have made use of three tier architecture. It is a type of software architecture which consist three tiers known as Presentation tier, Application Tier and Data Tier. Presentation tier is basically called template as it consists of user interface. They are accessed through web browser or web-based application which displays information and contents that are useful for a user. Application tier contains the functions business logic which drives to application core capabilities. It is written in c#, Python, php, C++. Lastly Data tier consists of the data storage system/database and data access layer. MySQL, Oracle, PostgreSQL are some examples of this system.

Benefits of using three tier architecture are:

1. It gives ability to make changes on working of one tier without making changes on another tier.
2. It allows reliability and independence of underlying services and servers.
3. It supports easy maintenance of coding and managing code base and business logic separately so that change in any tier won’t affect other.

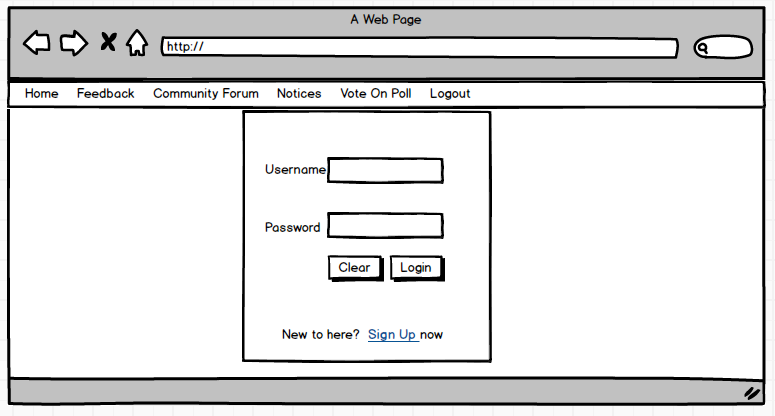


**UI Modelling**

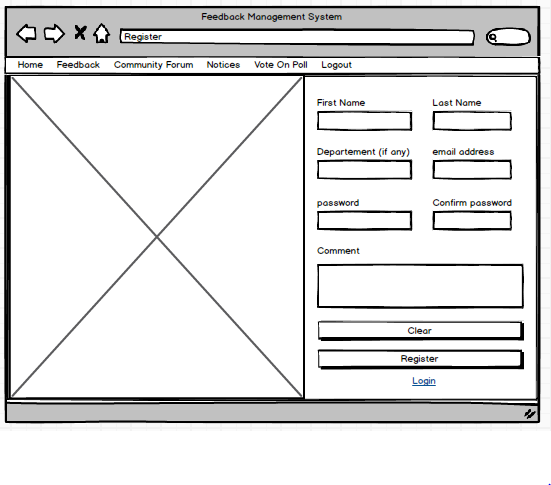
This is a type of modelling technique which shows how a User Interface will look like and how it would function. This is very important thing to do because if helps to get feedbacks from the users on the interface. So, if the client/User is not satisfied changes can be made upon user feedbacks.

For UI modelling I have used digital prototyping as it is incremental which means that we can do it again and again and another reason for using digital prototype is to minimize the use of traditional paper-based prototype.

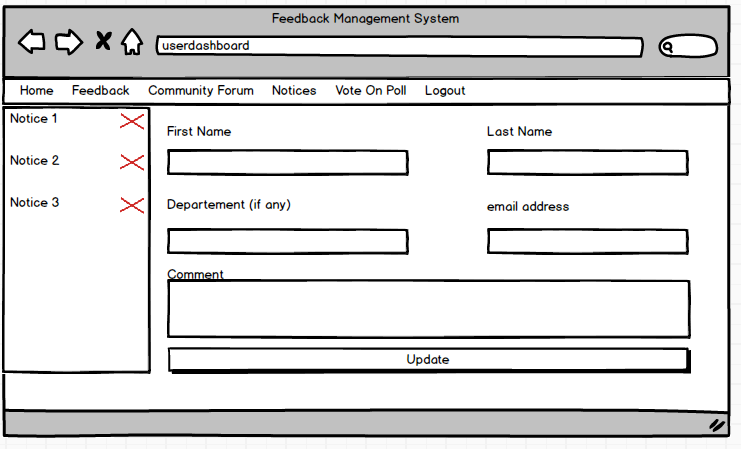
**Login**



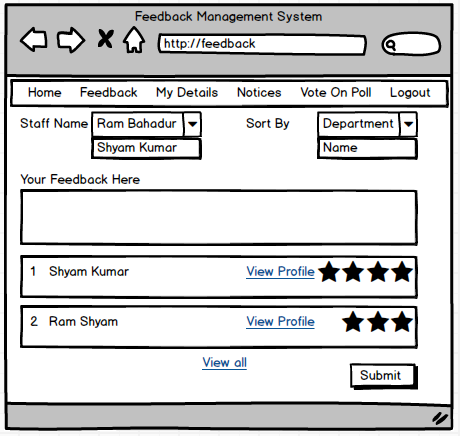
**Registration**



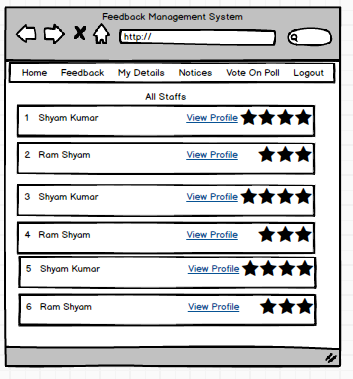
**Dashboard-User**



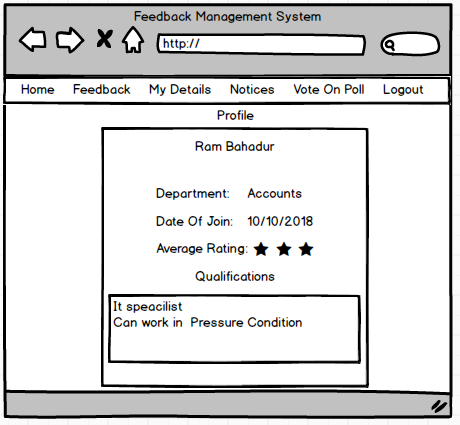
**Feedback**



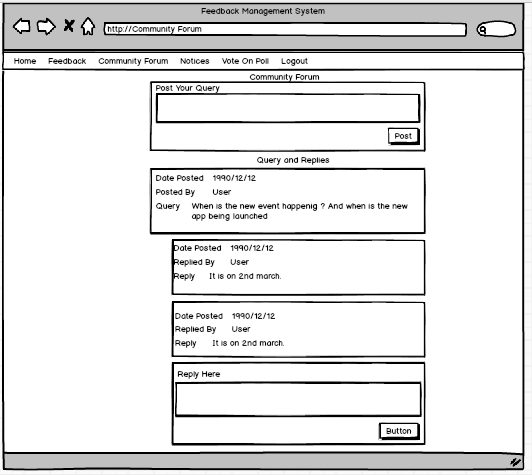
**Staff Profile**



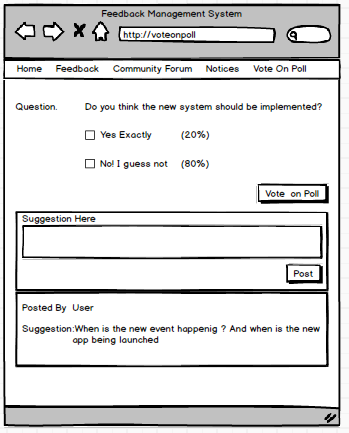
**Individual Staff Profile**



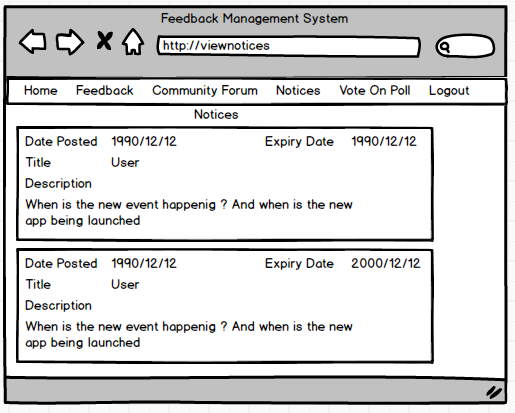
**Community Forum**



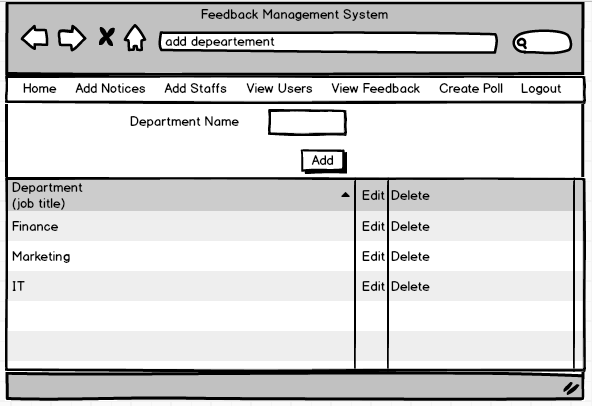
**Vote on Poll**



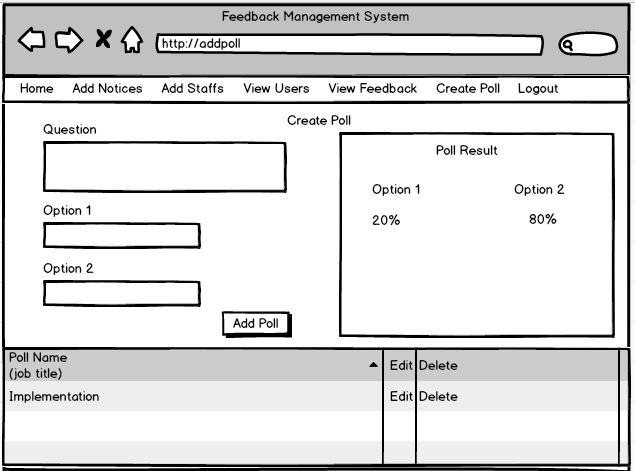
**View Notices**



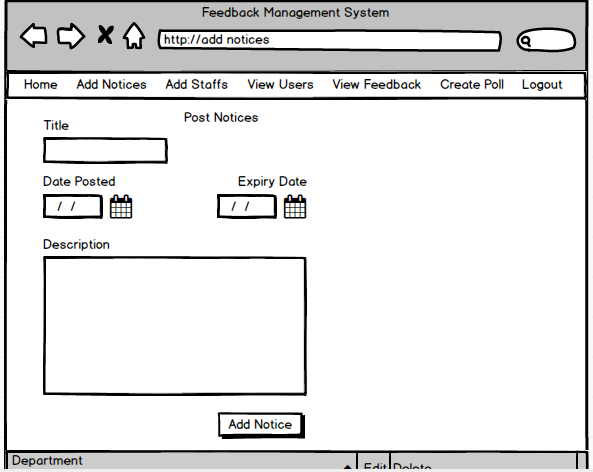
**Admin-Add Department**



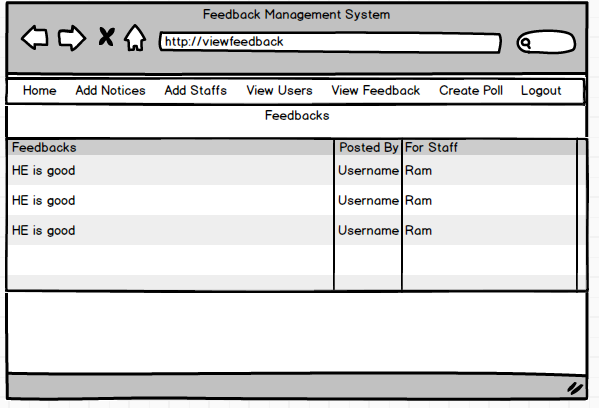
**Create Poll**



**Add Notices**



**View Feedbacks**



**Add Staffs**

